Multiresolution representations for surfaces meshes based on the vertex decimation method

Reinhard Klein

Wilhelm-Schickard-Institut, GRIS, Universität Tübingen, Auf der Morgenstelle 10 / C9, 72076 Tübingen, Germany, E-mail: reinhardgris.uni-tuebingen.de
http://www.gris.uni-tuebingen.de

The vertex decimation is a general mesh simplification approach. By successively removing vertices a hierarchy of different levels of detail (LOD) is generated during the simplification process. This paper enhances our 1996 mesh simplification algorithm with error control [25] to preserve some information about the normals of the original faces into the resulting simplified data. This enables us to build multiresolution models (MRM) which allow to control the normal deviation and to extract view dependent-adaptive lighting sensitive approximations of the original meshes.
Furthermore, the paper gives a general overview on MRMs generated by vertex removal algorithms. Where necessary material from our previous publications scattered in various conference proceedings and short papers (some of them hard to access) is included and extended with detailed algorithms and proofs. Although formulated for a vertex removal algorithm the results apply to other simplification algorithms and MRMs as well, namely to edge and triangle collapse algorithms.

CR Descriptors: I.3.3[**Computer Graphics**]: Picture/Image Generation, Display algorithms; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling *Curve, surface, and object representations, Object hierarchies* ; **Additional Key Words and Phrases:** Hierarchical approximation, model simplification, levels-of-detail generation, shape approximation, terrain modeling

1 Introduction

The visualization of large models of real-world objects, like cars, trains, airplanes, etc. is a major challenge in the context of virtual reality. In most sce-

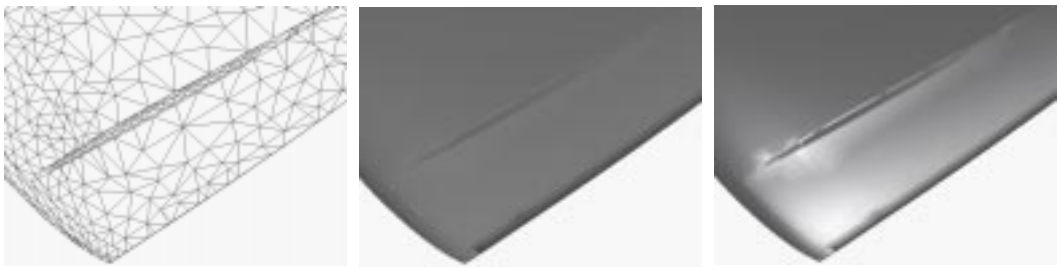

Fig. 1. The parts of the car bonnet are represented by the same level of detail shown on the left side. In contrast to the surface of the bonnet in the middle, the one on the right side contains specular components: artifacts become visible.

narios several such models have to be visualized and animated simultaneously. Examples are the optimization of the cabin of a train or the optimization of a driver's position in a car. In such a setting not only the particular train or car has to be visualized but also other cars, pedestrians, buildings, etc. Furthermore, due to new acquisition techniques and more elaborated modeling and animation techniques the size of the models is still growing. Despite the performance of modern graphics hardware the visualization of such scenarios cannot be realized in real-time without special techniques to reduce the number of triangles that are rendered.

Multiresolution modeling means maintaining objects at different level-of-detail (LOD) or approximation, where the LOD may be different in distinct areas of the object. And with respect to visualization it means to render the model at any given moment with minimal LOD sufficient to produce an image of reasonable quality. The quality of the resulting image not only depends on the geometric distance between the simplified object and the original model but also on the illumination of the object. The illumination depends on the different light sources and their positions, on the different materials, colors, textures, and normals of the object and last but not least on the shading model and shading technique which is used to render the model. A certain LOD may be sufficient to produce an image of reasonable quality if the object contains only diffuse surfaces that reflect radiation equally in all directions, but the same level of detail may be insufficient if the surface of the object also contains specular components. As an example Figure 1 shows a part of a car bonnet with different surface properties. The level of detail which is used is shown on the left side. The bonnet is rendered using hardware Gouraud-shading. In contrast to the bonnet on the right side, the middle one contains only low specular components. In this case the chosen level of detail is sufficient while in the other case broad artifacts are visible. Due to the hardware Gouraud shading in the area of the highlights a higher level of detail is necessary to get a sufficient image quality. A similar problem occurs near silhouettes w.r.t. the direction of light: minimal changes of the geometry may change a vertex from being lit and normally colored to pointing away from the light source and therefore being colored black. This leads to an inhomogeneous illumination in

Fig. 2. The model of the rim in the middle is approximated by the level of detail shown in the picture on the left side. In areas where the direction of light is nearly perpendicular to the normals of the rim-surface already minimal changes of the normals lead to broad visible artifacts. The picture on the right side is generated using a view-dependent triangulation that take care of the illumination as described below. The artifacts disappear.

areas where the normals of the object surface are nearly perpendicular to the direction of light and slightly perturbed due to the mesh simplification process, see Figure 2. Note that the illumination artifacts described here especially occur on smooth surfaces used in CAD-applications. In these applications the illumination plays an important role as a tool to visualize artifacts in the smoothness of the original surfaces. Although mesh simplification is necessary to speed up the visualization, artifacts in the illumination cannot be tolerated and must be avoided. Note that for objects with rough non specular surfaces like the brain surface reconstructed from CT-data or the surface of an animal used in an animation illumination artifacts are less important.

As shown by the examples the illumination artifacts can be avoided if in certain areas like the neighborhood of highlights or along the silhouettes a better approximation of the object, that means a higher level of detail, is used. The simplest way is to use a sufficient high level of detail for the complete object, which can guarantee the desired image quality, but in many cases it does not allow any simplification of the scene geometry. The situation might be improved a little bit, if during the simplification process aside from the geometric distance between original and simplified object also deviations between the normals are taken into consideration.

A further possibility is to take care of the illumination condition already in the simplification process and to simplify the object only in areas where no artifacts are expected. In general this approach leads to high quality images while reducing the number of triangles to be rendered, but in general mesh simplification algorithms are too slow for real-time applications. Nevertheless, there is some work on real-time simplification algorithms [37,28].

Like Xia et. al [36] and Hoppe [18], in this paper we use a two stage approach to overcome the problems described above: In an offline simplification process a view-independent multiresolution model (MRM) is generated. This model contains all the information about the geometric accuracy and about the deviations of the normals of the surfaces in all intermediate levels of detail which are needed to choose the appropriate level according to the viewing parameters, silhouettes and highlights. From this MRM we can extract view-dependent simplified meshes at variable resolution and render them with the desired image quality.

After a discussion of the previous work in section 2 we review in section 3 our simplification algorithm based on vertex removal and show how information about the deviation of normals of the simplified meshes can easily be incorporated. In the sections 4 and 5 we describe the construction of two different MRMs together with algorithms for the extraction of simplified meshes at variable resolution in more detail. The MRM described in section 4 is well suited for parameterized surface meshes and is based on a constrained Delaunay Triangulation (CDT). In addition to the already published results in two previous short papers [23,27] we will give a theoretical foundation as well as a detailed description and analysis of the extraction algorithms based on the CDT. The MRM described in section 5 is an extension of a MRM first presented by Puppo for terrain visualization [30] to free-form surfaces embedded in the three-dimensional space. In section 6 we discuss how the normal information can be used to avoid the extraction of most of the back-facing triangles. Furthermore, we discuss illumination controlled refinement based on the normal information. We end with some conclusions in section 7.

2 Previous work

In the literature a variety of techniques for the generation of multiresolution models has been proposed. To obtain a hierarchy of increasingly coarser approximations of the input object, mesh simplification algorithms are applied. Various techniques have been published that aimed to reduce surface complexity [14,29,20,34,5,25,2,19,17,32,33,35,7]. Most of these techniques simplify triangular meshes either by merging elements or by resampling vertices using different error criteria to measure the fitness of the approximated surfaces. For some surveys see also [3,15,8].

The most simple MRMs consist of a hierarchy of 5 to 10 different LODs of a given object. At run-time the perceptual importance of a given object in the scene is used to select its appropriate level of detail from the hierarchy [4,13,31]. The drawbacks of this approach are, that switching between two different levels of detail causes visual artifacts, and that of course no optimal

adaptive approximation is possible with discrete resolution steps.

De Floriani and Puppo [6] give a formal description of a Hierarchical Triangulation from which a selectively-refined generalized triangulation can be obtained. This generalized triangulation satisfies the given precision requirement at each point in the domain but cannot guarantee continuity between the triangles. Further vertices must be inserted into the triangulation to avoid the cracks between the triangles, but this may produce a mesh which violates the previously-satisfied precision requirement.

Simplification models that iteratively remove a vertex or collapse an edge or a triangle of a mesh implicitly define a hierarchy of different levels of detail of the simplified object. Each successive LOD differs from the previous one in the local vicinity of the removed vertex, edge, or triangle only. The advantage of this approach is that the difference between two successive LODs is relatively small. Changing between different levels does only cause small visual artifacts. MRMs that are built on this observation are for example the *progressive meshes* by Hoppe and the MRM proposed by Ciampalini et al. [2]. In addition, the progressive meshes allow for smooth geomorphing between all different levels of detail, resulting in a smooth transition. A shortcoming of these approaches is, that they do not efficiently support the co-existence of different levels of detail across different regions of the same object. The selective refinement described by Hoppe [17] is only a first step in this direction. As Hoppe pointed out, a stringent version of the refinement condition for selective refinement can cause the refinement to fail. Therefore, he suggests a less stringent condition, which necessarily introduces additional triangles. Unfortunately, in the proximity area of these additional triangles no bounds on the approximation errors between the refined triangulation and the original mesh are known.

Xia and Varshney [37] propose an algorithm based on a precomputed merge tree which supports several different levels of detail to co-exist across different regions of the object. The different levels of detail in different regions seamlessly merge with one another without introducing any cracks and discontinuities. In the following such a multiresolution model is called *variable multiresolution model*. In the context of mesh decimation algorithms based on vertex removal, a fundamental problem of the merge tree is, that the tree itself imposes a dependency on the refinement operations, that is artificial and not imposed by the problem. Only what the authors call dependencies is the necessary information. This information constitutes the directed acylic graph structure in our model and the model proposed by Puppo [30]. A further drawback of this approach is that the geometric error between simplified mesh and original mesh is only estimated. Every merge tree node stores an Euclidean distance for splitting a vertex to its child as well as the distance at which it will merge to its parents. In many cases this leads to an under-estimation of

the actual global error resulting in a poor approximation while in other cases these errors accumulate resulting in smaller reduction rates.

In our previous work [23,27] we showed how an adaptive MRM can be built for parameterized surfaces using a constrained Delaunay triangulation. The MRM is either computed by a mesh refinement-algorithm or by the mesh simplification-algorithm mentioned above [26,25]. In both cases, a global error between the simplified and original mesh is used to guarantee a sufficient approximation of the original mesh by the adaptive triangulation extracted from the MRM. Through the use of the constrained Delaunay triangulation the topology of the mesh is implicitly given and does not have to be stored explicitly. Especially for large models this is a significant advantage [27,21]. We will discuss this model in greater detail in Section 4.

In [30] Puppo introduces the multi-triangulation. It is based on a collection of fragments of plane triangulations arranged into a partially ordered set. Different decompositions of a domain can be obtained by combining different fragments from the model. A generalization of this model to higher dimensions is described in [12].

In [24] we extended the approach of Puppo to free-form surfaces embedded in the three-dimensional space and gave a dynamic algorithm that is able to refine a given triangulation in a certain area and to coarse it in another area. In section 3 we briefly review the main ideas of this approach.

3 Mesh simplification

Our simplification algorithm is a vertex decimation algorithm first described by [34]. It can be used for the simplification and construction of a MRM of arbitrary surface meshes. A simple modification of the algorithm allows to build up a special MRM for parameterized surfaces, see section 3.4.

3.1 The simplification algorithm

The mesh simplification algorithm always starts with the finest triangulation in 3D space approximating the original model surface. In the following this starting triangulation is denoted with Σ_M, indicating that it contains M vertices.

The simplification algorithm successively simplifies the starting triangulation by removing vertices from the current triangulation. All triangles adjacent to the removed vertex are removed from the current triangulation and the

resulting holes are retriangulated. This is done until no further vertices can be removed from the simplified triangulation without exceeding a predefined modified one-sided Hausdorff distance between the original triangulation and the simplified one, see section 3.1.1. If after removing some vertices the simplified triangulation contains m vertices it is denoted by Σ_m.

3.1.1 The modified one-sided Hausdorff-distance

The Euclidean distance between a point x and a set $Y \subset \mathbb{R}^n$ is defined by

$$d(x, Y) = \inf_{y \in Y} d(x, y),$$

where $d(.,.)$ is the Euclidean distance between two points in $\mathbb{R}^n$. Using this definition we can define the one-sided Hausdorff distance $d_E(X, Y)$ *from* a set X *to* a set Y by

$$d_E(X, Y) = \sup_{x \in X} d(x, Y). \tag{1}$$

It doesn't define a distance function on the set of all sets of $\mathbb{R}^n$, because it is not symmetric. That means that in general $d_E(X, Y) \neq d_E(Y, X)$.

If the one-sided Hausdorff distance $d_E(\Sigma_M, \Sigma_m)$ between the original triangulation Σ_M and the simplified triangulation Σ_m is less than a predefined error tolerance ϵ, then

$$\forall x \in \Sigma_M \text{ there is a } y \in \Sigma_m \text{ with } d(x, y) < \epsilon.$$

Therefore, this one-sided Hausdorff distance between the original and the simplified triangulation is the one a user would intuitively think of. But in some cases the unsymmetry of the one-sided distance leads to problems. This can either happen near to the borders of the original mesh or at parts of the mesh that resemble to the border, in the sense that the angle between adjacent triangles along a common edge is very small. An example is the concave blade of a sickle, see Figure 3. One way to overcome this problem is to use the full *Hausdorff distance* instead of the one-sided distance. It is defined by

$$d_H(X, Y) = \max(d_E(X, Y), d_E(Y, X)). \tag{2}$$

In contrast to the one-sided Hausdorff distance it is symmetric and we have

$$d_H(X, Y) = 0 \Longleftrightarrow X = Y.$$

Unfortunately its computation is too expensive. Therefore, we make a compromise between speed and accuracy. In addition to the one-sided Hausdorff distance $d_E(\Sigma_M, \Sigma_n)$ between the original and a simplified triangulation we

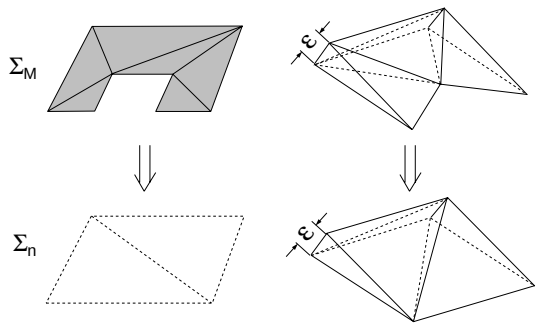

Fig. 3. Left: The original nonconvex triangle mesh Σ_M is contained within a plane. The simplified mesh Σ_m consists of a triangulation of the convex hull of the original mesh and therefore the original mesh is contained in the simplified mesh. Right: The same situation in 3D. $\forall x \in \Sigma_M T$ there is a $y \in \Sigma_m$ with $d(x, y) < \epsilon$ but we do not have $\forall y \in \Sigma_m S$ there is a $x \in \Sigma_M$ with $d(x, y) < \epsilon$.

also compute the one-sided Hausdorff distance between the border of the simplified mesh and the original mesh and call this distance *modified one-sided Hausdorff distance*. In such a way for a given ϵ an accurate approximation of the borders of the original mesh by the border of the simplified mesh can be guaranteed while for every inner point x of the original triangulation Σ_M there is always a point y of the simplified triangulation Σ_n with $d(x, y) \leq \epsilon$.

3.1.2 The priority queue

The priority queue is one of the central features of the mesh simplification algorithm. For every single vertex v_i of the simplified mesh an error value ϵ_i is computed and stored. This value describes the *potential error*, that is the modified one-sided Hausdorff distance that would occur between the simplified and the original mesh if the vertex would be removed. At the beginning of the algorithm this error value is computed for every vertex in the original mesh. The vertices of the original mesh are then sorted into a priority queue according to their error-value. Afterwards, in each simplification step we successively eliminate the vertex with the smallest potential error on top of the priority queue. If a vertex is actually removed from the current simplified triangulation the potential errors of all its neighbor vertices are recomputed and the priority queue is updated. There are two cases where the removal of a vertex would not make sense: First, so-called complex vertices (see [34]) and second, vertices for which the retriangulation of the resulting hole may lead to topological problems. These situations are detected by topological consistency checks, see [35]. In both cases the potential error of the vertex is set to infinity.

The priority queue in combination with a global geometric error guarantees that the simplified triangulation Σ_m – obtained after removing the first $M - m$ vertices from the priority queue – approximates the original triangulation with a global geometric error ϵ_m. This error can also be considered as a *local error*: If

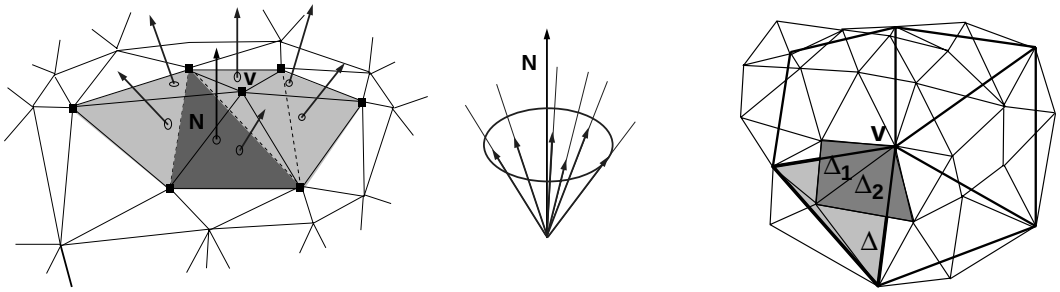

Fig. 4. Left side: For each triangle of the simplified triangulation a maximum normal deviation around the normal of the triangle is stored. This defines the so-called cone of normals. Right side: The triangles Δ_1 and Δ_2 are the only two triangles incident to v whose projection onto the plane defined by triangle Δ intersects Δ. Therefore, only the normals of these two triangles incident to v contribute to the normal deviation over the triangle Δ.

the next vertex v_{m+1} is removed also its adjacent triangles are removed and the resulting hole is retriangulated. The global error between the original and the simplified mesh changes to ϵ_{m+1}. If the current global approximation error ϵ_{m+1} is greater than ϵ_m it occurs between the new triangles of the retriangulated area and the original triangulation, since only the triangles in the neighborhood of v_{m+1} are changed.

3.2 Generating information about the normals

To get information about the deviation of the normals, the correspondence map between the vertices and triangles of the original triangulation and the triangles of the simplified triangulation used for the distance calculations [25] is exploited. For a given triangle $\Delta(v_{k_1}, v_{k_2}, v_{k_3})$ of the simplified triangulation with normal n this map delivers all vertices v_i of the original triangulation that are closer to Δ than to any other triangle of the simplified triangulation. For each of these vertices v_i the incident triangles Δ_{ij} and their normals n_{ij} are determined. Then a maximum deviation angle

$$\cos(\delta) = \max_i \max_j \frac{n \cdot n_{ij}}{\| n \| \| n_{ij} \|} \tag{3}$$

is computed. To avoid unnecessary large deviation angles in this step the triangles incident to the vertices v_{k_i} of the triangle itself are handled differently. Only those incident triangles are taken into account for which the projection onto the plane determined by the three vertices $v_{k_1}, v_{k_2}, v_{k_3}$ intersect the triangle Δ, see Figure 4. For each triangle of the simplified triangulation this maximum angle defines a cone of maximal normal deviation around the normal of the triangle, see Figure 4.

Fig. 5. On the left side the mesh of the mandible produced by the marching cube algorithm. In the middle a simplified mesh. On the right side mesh of the mandible is refined around the left tooth. The area of interest was defined interactively.

3.3 Retriangulating the resulting hole

The resulting hole can be retriangulated in different ways. A closer look on different retriangulation strategies shows that from the topological point of view the edge collapse approach can be considered as a special retriangulation technique for the resulting hole in the simplification algorithm [24]. The retriangulation of the resulting hole determines the quality of the resulting simplified triangulations, as well as the reduction rates that are achievable by the algorithm. While using a CDT optimizes the shape of the triangles in the simplified triangulations this technique does not deliver high reduction rates for certain objects like a cylinder. In such cases much better reduction rates can be achieved using the edge collapse technique or the geometric optimization technique described in [24]. Nevertheless, there are cases like the 'mandible' in Figure 5. In this example we want to do finite-element computations and are interested in good aspect ratios of the triangles. In this case the results using a Delaunay triangulation are superior to the other techniques.

3.4 The simplification algorithm for parameterized surfaces

For parameterized surfaces a more compact MRM is possible than in the general case of arbitrary surface meshes. This special MRM is based on the use of a Delaunay-triangulation in parameter space. In order to build up such a MRM the simplification algorithm has to be modified. In order to get the starting triangulation Σ_M a preprocessing step is performed.

3.4.1 The preprocessing step

In this step the given parameterized surface $f\colon \mathbb{R}^2 \supset \Omega \longrightarrow \mathbb{R}^3$ is presampled in such a way, that the Constrained Delaunay triangulation (CDT) Σ_M of the pre-sampled vertices $(u_j, v_j) \in \Omega\ j = 1, \cdots, M$ in the parameter-domain delivers a piecewise linear approximation f_{Σ_M} of the surface with a maximum parametric approximation error $d_a(f, f_{\Sigma_M}) = sup_{(u,v)\in\Omega}||f(u,v) - f_{\Sigma_M}(u,v)|| < \eta$.

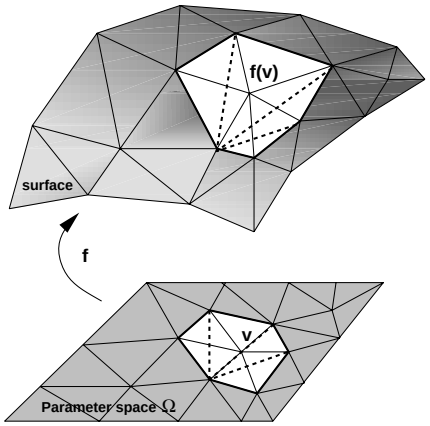

Fig. 6. After removing vertex $f(v)$ from the actual approximation of the surface a CDT of the corresponding resulting hole in parameter space is computed.

The following simplification step works on this piecewise linear approximation f_{Σ_M}. The CDT Σ_M in parameter space, which corresponds to the piecewise linear approximation f_{Σ_M} of the original surface defines the original model. In order to get a unique Delaunay triangulation we assume that no four sample points lie on a circle. Otherwise, the sample points are perturbed slightly.

3.4.2 Retriangulation of the resulting hole

In the case of general surface meshes the retriangulation of the resulting hole after removing a vertex is done by projecting the vertices of the border polygon of the resulting hole onto a special plane in 3D. Instead of this, in the case of parameterized surfaces the resulting hole is retriangulated in the corresponding CDT in parameter space. To end up this, a CDT of the polygon, defined by the resulting hole has to be computed, see Figure 6.

In such a way, a hierarchy $\Sigma_M, \ldots, \Sigma_m$ of successively coarser CDT's in parameter space is generated. The corresponding sequence of triangulations $f_{\Sigma_k},\ m \leq k \leq M$ in 3D-space defines different LODs of the original surface.

3.4.3 Parametric distance versus Hausdorff distance

Instead of the modified one-sided Hausdorff distance, in the case of parameterized surfaces we could as well use a parametric distance. By using a parameterized distance the correspondence map between the original and the simplified triangulation is given implicitly and needs not be stored during the simplification process. Nevertheless, it is worthwhile to mention that for *any* parameterized surface $f : \mathbb{R}^2 \supset \Omega :\longrightarrow \mathbb{R}^3$ that is approximated by a piecewise

linear surface $f_\Sigma :: \mathbb{R}^2 \supset \Omega :\longrightarrow \mathbb{R}^3$ we always have

$$d_H(f, f_\Sigma) \leq ||f - f_\Sigma||_\infty = \sup_{(u,v)\in\Omega} ||f(u,v) - f_\Sigma(u,v)||.$$

For this reason, using the Hausdorff distance for error measurements results in higher reduction rates for the same error tolerance.

4 The MRM based on the Delaunay hierarchy

From the coarsest CDT Σ_m and the sequence of removed vertices $(v_{m+1}, v_{m+2}, \ldots, v_M)$ transforming Σ_m into the triangulation Σ_M at full resolution all intermediate unique Delaunay triangulations $\Sigma_m, \Sigma_{m+1}, \ldots, \Sigma_M$ can be recomputed through simple point insertion. Therefore, the multiresolution representation of a surface patch requires a coarse Delaunay triangulation Σ_m of the domain in the XY- plane, the sequence of points transforming Σ_m into the triangulation Σ_M at full resolution and the corresponding simplified triangulation Σ_n. The topology of the triangulation and the hierarchy of the different levels of detail is implicitly given by the use of the CDT and *does not have to be stored* explicitly. To each vertex v_n of the sequence the global approximation error ϵ_n between the original triangulation Σ_M and the intermediate triangulation Σ_n is stored. Since the model does not explicitly contain triangles, their normal cones cannot be stored together with the triangles. Instead, we store the normal cones containing the normal cones of all triangles belonging to a retriangulated hole. This cones describe the maximum deviation of the normals from a average normal of the triangles which are incident to the removed vertex v_n.

Avoiding the storing of the topology and the hierarchy of the different levels of detail of the multiresolution model leads to a massive reduction of the total storage costs. This is the great advantage of this approach compared to other MRMs. To discuss the total storage costs, we distinguish between

(i) arbitrary parameterized $3D$-Surfaces defined by a function $f : \mathbb{R}^2 \supset \Omega \longrightarrow \mathbb{R}^3$ and
(ii) explicit parametrized surfaces like digital terrain surfaces defined by a function $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$.

In both cases the coarsest triangulation Σ_m in parameter space and the sequence of vertices $v_m, \ldots, v_M$ have to be stored. The storage cost for the topology of the initial coarsest triangulation can be considered to be constant.

In the first case in addition to the $2D$-vertices the function f must be stored, in order to recompute the locations of the vertices in $3D$-space. The amount

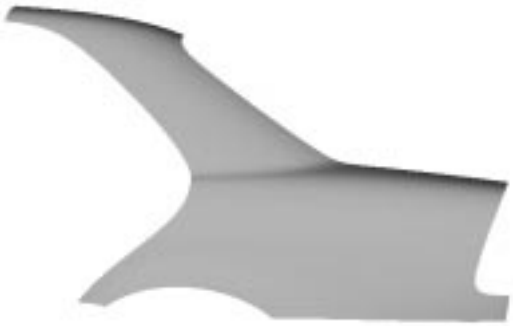

Fig. 7. This side part of the outer skin of a car is defined by 9 NURBS patches with a total of 3274 control vertices. The original approximation of the surface contains about 50.000 vertices.

of data to store the function f depends on the representation of f. In general the storage costs needed to store f are much less than the storage costs needed to store the locations of the corresponding $3D$-vertices $f(v_m), \ldots, f(v_M)$: As an example we consider the part of the outer skin of the car in Figure 7. It is defined by 9 trimmed NURBS-patches with a total of 3274 $3D$-control vertices. About 10 kB storage are necessary to store the control vertices of the patches. Instead of storing the trimming curves of the patches, it is sufficient to store for each vertex the number of the patch. If we assume that we spend 1 Byte to identify the patches this leads to a storage cost of 60 kB to store the Delaunay hierarchy. This are about 1.2 Byte per vertex.

In the second case of explicit parameterized surfaces to each vertex in parameter space only its z-value must be stored. No additional memory is needed to store the connectivity.

4.1 Extracting triangulations with a constant approximation error

Extracting triangulations with a constant approximation error all over the object is much simpler than extracting triangulations at variable resolution. For many practical applications this is enough. For example, compared to the complete visible part of a complex CAD-model the size of one patch is relatively small. In this case it is sufficient to extract each patch with its own constant approximation error. This approximation error depends on the viewing parameters and the bounding box of the patch in 3D-space and can easily be updated after a change of the camera position.

To extract a mesh with constant approximation error over the complete object we start with the coarsest triangulation Σ_m in parameter space and insert stepwise vertices into the corresponding CDT until the geometric error $G(v)$ of the inserted vertex v is less than ϵ. If we have already extracted a triangulation

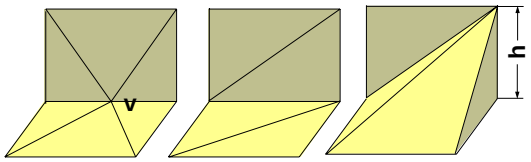

Fig. 8. After removing vertex v the triangulation shown in the middle exactly approximates the geometry of the object on the left side. If an edge is flipped the approximation error of the resulting triangulation shown in the right Figure depends on the height h of original model and cannot be controlled without explicit distance computations.

that guarantees a smaller error tolerance than ϵ it is not necessary to start again with the coarsest triangulation. Instead of this, vertices are removed from the triangulation, beginning with the vertex with the highest index (the one which was inserted last).

4.2 Extracting a triangulation at variable resolution

To get a CDT at variable resolution one might be tempted to leave out some vertices of the hierarchy. But using this approach the resulting triangulation may contain triangles that do not belong to any intermediate triangulation $\Sigma_m, \ldots, \Sigma_M$. For such a triangle no approximation error can be guaranteed! This is shown in Figure 8. The triangulation in the middle approximates the object on the left side exactly. Changing this triangulation causes an uncontrolled approximation error that depends on the object geometry and cannot be controlled without explicit distance computations.

Therefore, in order to take advantage of the precomputed approximation errors we need to use triangulations which are consistent with the precomputed CDT.

Definition 4.1 *An adaptive triangulation* $\Sigma_{V'} = \mathcal{T}(V')$ *with* $V' = \{v_1, \ldots, v_m, v_{i_1}, \ldots, v_{i_j}\} \subset V = \{v_1, \ldots, v_M\}$ *is called consistent with respect to the hierarchy* $\Sigma_m, \ldots, \Sigma_M$*, if for all triangles* $\Delta \in \Sigma_{V'}$ $\exists\ l,\ m < l \leq M$*, in such a way that* $\Delta \in \Sigma_l$*. A triangle* Δ *of a triangulation is called* valid *with respect to the hierarchy* $\Sigma_m, \ldots, \Sigma_M$ *if it belongs to a triangulation which is consistent with respect to the hierarchy.*

The following notation is adopted from Puppo [30]. We assume that for each triangle Δ in the multiresolution model a Boolean condition $c()$ is defined. $c(\Delta)$ is true if and only if the resolution of Δ is acceptable. Similarly, the notation $c(\mathcal{T})$ means that all triangles of a triangulation $\mathcal{T}$ satisfy $c()$.

We consider the following problem:

Given a multiresolution model as described above, extract the smallest tri-

angulation $\mathcal{T}$ consisting of triangles from possibly different levels of detail, so that $c(\mathcal{T})$ is true.

In the following we show how based on the ordered list $v_m, \ldots, v_M$ of removed vertices such a triangulation can rapidly be computed without storing additional information.

Let us briefly recall the concept of the influenced area of a vertex v, see [11].

Definition 4.2 *Given a CDT $\mathcal{T} = \mathcal{T}(V, E)$ where V is the set of vertices and E is the set of constrained edges and a vertex $v \notin V$. The collection $\mathcal{T}_v$ of the triangles $\Delta \in \mathcal{T}$ such that the circumcircle of Δ contains v and v is visible from all the vertices of Δ is called influenced area of the vertex v w.r.t. the triangulation $\mathcal{T}$. The polygon formed by the external edges of the triangles in $\mathcal{T}_v$ is called the influence polygon P_v of vertex v w.r.t $\mathcal{T}$.*

The following two Lemmas can be found in [11].

Lemma 4.3 *The insertion of a new vertex v in a CDT $\mathcal{T}$ modifies a triangle Δ of $\mathcal{T}$ if and only if Δ is in $\mathcal{T}_v$.*

Lemma 4.4 *Let $\mathcal{T}'(V', E')$, $V' = V \cup \{v\}$ be the CDT obtained from $\mathcal{T}(V, E)$ by the insertion of v. Let $\mathcal{T}'_v$ be the triangulation of the influenced polygon P_v of v obtained by connecting v with the vertices of P_v, then $\mathcal{T}' = \mathcal{T}'_p \cup \mathcal{T} - \mathcal{T}_p$.*

The following lemma follows directly from the construction of the Delaunay hierarchy:

Lemma 4.5 *If all vertices v_i with corresponding maximum geometric error $\epsilon_i > \epsilon$ are inserted into the CDT the resulting triangulation is consistent with respect to the hierarchy and the global geometric error is guaranteed to be less than ϵ.*

Lemma 4.6 *Suppose that a triangle $\Delta(v_i, v_j, v_k)$ of a reduced CDT T at variable resolution in the domain is given. If the circumcircle of $\Delta(v_i, v_j, v_k)$ does not contain any vertices with corresponding insertion index less than $l = \max(i, j, k)$, the triangle Δ belongs to the intermediate triangulation Σ_l and approximates the surface up to an approximation error $G(p_l)$, see Fig. 9.*

Proof: Successively removing all vertices v_n with index $n > l$ from the CDT and recomputing the resulting CDT will not destroy the triangle $\Delta(v_i, v_j, v_k)$. Also inserting all vertices with index less than l not already contained in the triangulation would according to the lemmas 4.3 and 4.4 not destroy the triangle $\Delta(v_i, v_j, v_k)$, since according to the assumption none of these vertices is contained in the circumcircle of $\Delta(v_i, v_j, v_k)$ and therefore, Δ is not contained in any of their influenced areas. This ends the proof.

The following corollary and theorem follow immediately from the lemma.

Corollary 4.7 *Assuming that only consistent triangulations are generated, a valid triangle $\Delta(v_i, v_j, v_k)$ of a CDT remains in the triangulation until the point with lowest index r greater than $l = \max(i, j, k)$ contained in the circumcircle of $\Delta(v_i, v_j, v_k)$ is inserted into the triangulation.*

Theorem 4.8 *Let $\Sigma_{V'} = \mathcal{T}(V')$ with $V' = \{v_1, \ldots, v_m, v_{i_1}, \ldots, v_{i_j}\} \subset V = \{v_1, \ldots v_M\}$ be a consistent triangulation w.r.t. the hierarchy $\Sigma_m, \ldots, \Sigma_M$ and $v_l \ni \Sigma_{V'}$. The CDT $\Sigma_{V''} = \mathcal{T}(V' \cup \{v_l\})$ is consistent w.r.t. $\Sigma_m, \ldots, \Sigma_M$ if and only if in the influenced area $\mathcal{T}_{v_l} \subset \Sigma_{V'}$ of v_l there is no vertex v_n, $n < l$. In this case the influenced area of v_l w.r.t. the current triangulation is the same as the influenced area of v_l w.r.t. the triangulation Σ_l.*

This leads to the following algorithm to compute a triangulation at variable resolution:

(i) Start with the coarsest CDT Σ_m.
(ii) Put each triangle $\Delta \in \Sigma_m$ onto a stack U.
(iii) While U is not empty
  (a) Fetch the first unmarked triangle Δ from the top of U.
  (b) If $c(\Delta)$ is not true
  determine $r := \max(v_i, v_j, v_k)$, where v_i, v_j, v_k are the vertices of Δ and find the vertex v_r with smallest index s, so that $s > r$ and v_s is contained in the circumcircle of Δ.
  Insert v_s into the CDT and correct the resulting triangulation.
  Put all triangles that are generated during the correction step onto the stack U and mark all triangles that are deleted during this step.
  If no such vertex v_r is found the triangle Δ is contained in original triangulation Σ_M and cannot be refined further. In this case the conditions are ill-posed and even Σ_M cannot satisfy them.

4.2.1 The correction step

The crucial part of the above algorithm is the correction step. In general, the triangles incident to the inserted vertex are not consistent w.r.t. the hierarchy $\Sigma_m, \ldots, \Sigma_M$, since the influenced area of the vertex v_r w.r.t. the triangulation before inserting v_r in general does not coincide with the influenced area of the vertex v_r w.r.t. to the triangulation Σ_r. But in order to get a consistent triangulation w.r.t. $\Sigma_m, \ldots, \Sigma_M$ according to the theorem 4.8 these two influenced areas must coincide. Therefore, we have to insert all vertices of the triangulation Σ_r that belong to the influenced area of v_r w.r.t. Σ_r. This is realized by the following correction algorithm:

(i) Put v_r on an empty stack V.

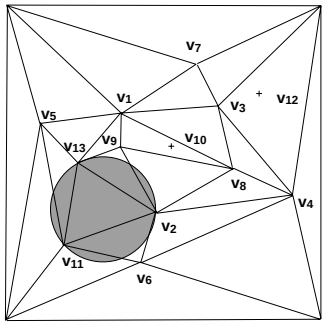

Fig. 9. Since the circumcircle of the triangle $\Delta(v_{13}, v_{11}, v_2)$ does not contain any other vertex with corresponding insertion index less than $13 = \max(i, j, k)$ the insertion of the points v_{10} and v_{12} will not change the triangle. Therefore, the elevation surface is approximated over the triangle $\Delta(v_{13}, v_{11}, v_2)$ up to a maximum approximation error ϵ_{13}.

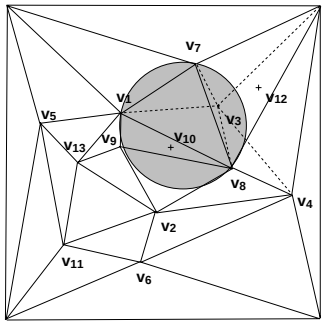

Fig. 10. Since the circumcircle of the triangle $\Delta(v_1, v_8, v_7)$ does contain point v_3 this triangle does not belong to an intermediate triangulation of the multiresolution model. Therefore, in the correction step vertex v_3 is inserted into the triangulation. After the insertion of v_3 the complete triangulation is valid.

(ii) While V is not empty fetch the first vertex v with index l from the stack V.
(iii) Reconstruct the influenced area of v_l w.r.t. the triangulation Σ_l.
(iv) Put all vertices inserted during the reconstruction step onto the stack V and all triangles generated during this step on the stack U. Mark all triangles deleted during the reconstruction step.
(v) Go to step 2.

Reconstruction of the influenced area To reconstruct the influenced area of vertex v_l w.r.t. Σ_l we first consider the triangles of the current triangulation that are incident to v_l and check for each of these triangles if their circumcircle contains a vertex with index $v_n, n < l$ not already contained in the current triangulation. If for none of the incident triangles such a vertex is found, we are done, else we distinguish two cases to find an edge incident to v which is contained in the triangulation Σ_l:
1.) If all triangles incident to v contain vertices with indices less than l, we

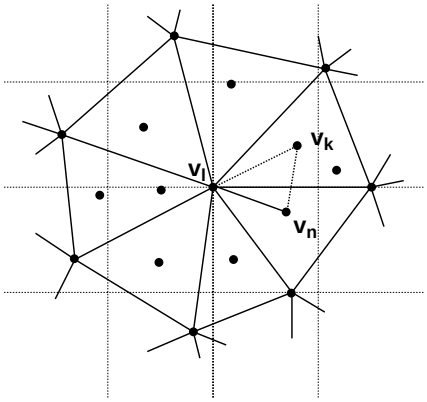

Fig. 11. After we found a valid edge (v_l, v_n) we search for the third vertex v_k of a valid triangle $\Delta(v_l, v_n, v_k) \in \Sigma_l$.

choose the one with smallest distance to v which is visible from v, since the edge between v and this vertex must be contained in Σ_l.
2.) If there exists a triangle which does not contain a vertex with an index less than l in its circumcircle we can choose one of its edges since these edges are contained in Σ_l.

After we found an edge $e = (v_l, v_n)$ of Σ_l we search for the third vertex v_k of the triangle $\Delta \in \Sigma_l$ containing the edge e. To complete this, we use parts of the algorithm described by Fang and Piegl [9,10], see Figure 11. Despite of the worst case complexity of this algorithm is $O(l)$ it has been shown by Fang and Piegl that using a regular grid in real applications this vertex can be found in constant time. This is also confirmed by our own experience.

Properties of the algorithm From these facts the following properties of the algorithm can be concluded:

- The resulting triangulation at variable resolution is consistent w.r.t. the hierarchy $\Sigma_m, \ldots, \Sigma_M$.
- The resulting triangulation is minimal among the triangulations fulfilling the conditions $c(\mathcal{T})$ and being consistent w.r.t. the hierarchy $\Sigma_m, \ldots, \Sigma_M$.
- Assuming that the insertion of vertices into the CDT can be done in linear time, the time complexity of the algorithm is linear in the number of inserted points, and therefore in the number of resulting triangles.
- Altough the algorithm saves storage capacity at the expense of computing power it is fast. We found that for real-world applications, due to the smooth changes of the camera, only a few vertices have to be inserted or removed from the triangulation. As shown in our previous work [23] in one of our first implementations the time for inserting 150 vertices was about 200 msec on an Indigo with R4400 Processor at 150 MHz.

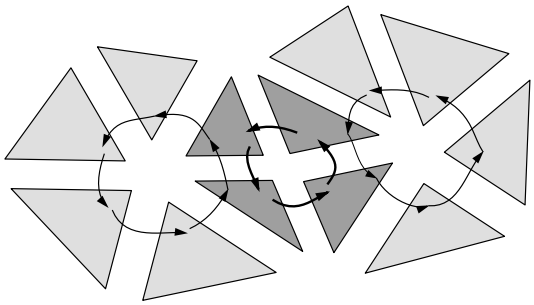

Fig. 12. To store the bottoms and the ceilings of the fragments acjacent triangles are organized in a linked list. In such a way each triangle belongs to the linked list representing its bottom fragment and the linked list representing its top fragment.

5 The explicit MRM

In the following we describe the data-structure of the multiresolution model which can be used for parameterized as well as for non-parameterized surfaces. In each vertex removal step two sets of triangles are involved: the triangles incident to the removed vertex and the new triangles retriangulating the resulting hole. This two sets form a fragment. The set of removed triangles is called ceiling, the set of triangles of the retriangulation is called floor. Each simplification step and its inverse is uniquely described by the removed vertex and the corresponding fragment. Therefore, storing the vertices $v_m, \ldots, v_M$ and the corresponding fragments $f_m, \ldots, f_M$ each intermediate triangulation $\Sigma_n, m \leq n \leq M$ can be reconstructed from the coarsest triangulation Σ_m by reversing the simplification process, that is by successively replacing the triangles in the floor of fragment f_i by the triangles in its ceiling. Note that reversing the order of the fragments between simplification and reconstruction process guarantees that in each of the reconstruction steps the current triangulation Σ_i contains all triangles of the floor of the next fragment f_{i+1}. A simple example of a multiresolution model containing only 4 fragments is given in Figure 13.

In order to reconstruct a consistent triangulation at variable resolution the triangles in the ceiling of a fragment can only be inserted if the triangles in it's floor are already contained in the triangulation. If not, we have to search and insert them first. This process continues recursively to build up the supporting fragment hierarchy. For this purpose each triangle stores two fragments, one below containing the regarded triangle in its ceiling, the other above, containing the regarded triangle in its floor. The first one is needed to find the corresponding floor triangles and is called top fragment. The second fragment is used for further refinement and is called bottom fragment. The fragments themselves can be stored implicitly: each triangle contains one pointer to an adjacent triangle in it's bottom fragment and one pointer to an adjacent triangle in it's top fragment, see Figure 12.

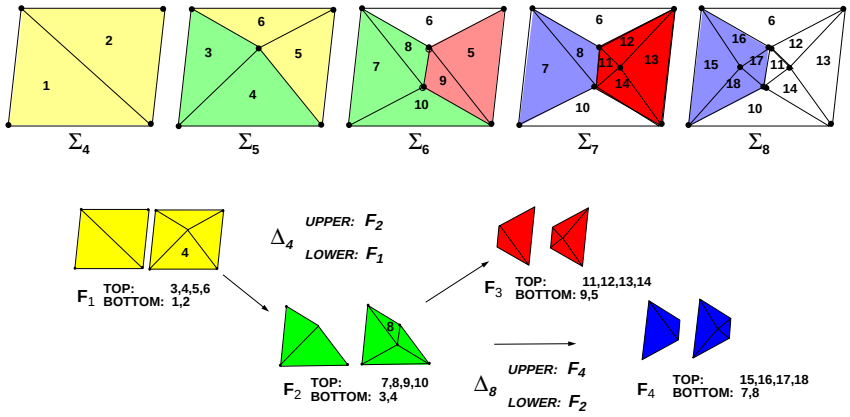

Fig. 13. A fragment contains indices to the triangles of the bottom and of the ceiling. On the other hand each triangle stores two pointers to the fragments containing it.

We use the following data structure for the explicit multiresolution model:
Struct MRModel
  List <**Triangle**> triangles;
  double normal_deviation;
  List <**vertex**>
  double error;

Struct Triangle
  TriangleIndex topFragment;
  TriangleIndex bottomFragment;
  VertexIndex vertexIndex[3];

Note that this data structure is very similar to the one first proposed by Puppo in [30] for multiresolution models of triangulations in the plane. Using this data structure Puppo was able to present the first algorithm working on MRMs of arbitrary triangulations in the plane correctly merging triangles from different levels of detail into one triangulation. This algorithm always starts from the coarsest triangulation and refines the triangulation as needed. In contrast to our data structure in the data structure of Puppo the fragments are stored explicitly resulting in higher storage requirements. For practical usage we improved Puppo's extraction algorithm in two ways: First we allowed the refinement to start with the already extracted triangulation rather than from scratch. Secondly we developed a second algorithm that reverses the refinement algorithm and allows us to selectively coarsen an already extracted triangulation. In [24] we described the implementation of these algorithms. Using these two algorithms the frame to frame coherence of the meshes can be exploited in real-time fly-troughs or animations which would otherwise be impossible. In the algorithms we use a function $c(\Delta)$ to determine if a triangle Δ sufficiently approximates the original triangulation or if further refinement is needed.

5.1 Relation between the two models

The influenced areas of the CDT are exactly the fragments of the multi-triangulation, see Figure 13. The main difference between the two models is, that in the multi-triangulation these dependencies among the fragments are stored explicitly, while using the CDT, these dependencies are given implicitly. As shown in Section 4 the upper fragment of a triangle (influenced area of the vertex with smallest index contained in the circumcircle of the triangle) and the lower fragments (influenced areas) of the triangles belonging to this fragment are reconstructed during the extraction algorithm. The only difference between the two algorithms is the order in which the fragments are visited.

5.2 Compression of the explicit MRM

Due to the storage capacity needed to store all the triangles and fragments of the explicit MRM on most workstations MRMs of large models cannot be kept in main memory. Therefore interactive visualization of the models is not possible. Note that the explicit MRM contains approximately $N_{MR} \approx 3N_{orig} - 2N_{red}$ triangles, where N_{orig} is the number of original triangles and N_{red} is the number of triangles of the simplified mesh at its coarsest level. For visualization purposes this is the main drawback of the explicit model. For this reason we developed a storage-optimized model for the MRM [22]. The main idea of this approach is to build equivalence classes for the fragment topology. For example all five possible triangulations of a convex pentagon can be obtained through simple rotations from only one triangulation. Representatives of these equivalence classes are stored as rules that describe how the bottom of the represented fragment is retriangulated. Now, for every fragment of the triangulation the number of its representative and its anchoring with the underlying triangulation is stored. The dependencies between the fragments are realized as linked lists. In this way the complete MRM together with all dependencies can be stored with about 24 Bytes per Vertex. Note that all information needed by the extraction algorithms described above are contained in the compressed model. This leads to a about 6 times smaller MRM than the comparable ones described by Hoppe [18] or Xia and Varshney [36].

6 Applications using the two-stage approach

After computing a MRM, either using the CDT, or storing the fragments explicitly, the extraction of variable levels of detail from the MRM with guar-

anteed approximation error in real-time is feasible. In the following we show how in addition to the view-dependent geometric approximation also a view-dependent illumination can be realized. To achieve this goal in the condition $c(\Delta)$ described in section 3 which decides if the area around a triangle should be refined further, we must take care of the current illumination conditions. This is possible since the cone of normals of the triangle stored in the MRM tells us about the maximum normal deviation of the model at finest resolution in the area of the triangle. In any of the following cases the refinement can be stopped if the diameter of a triangle is less than one pixel.

6.1 Avoiding to extract triangles on the back-side

Common rendering systems like OpenGL [16] allow to specify that triangles on the backside of the objects are not sent to the hardware graphics pipeline accelerating the rendering process by up to a factor of two. But using a MRM it is even not necessary *to extract* backside triangles of the object from the MRM. Assume that the triangle Δ with normal N and normal deviation angle α and the direction E to the observer is given. If

$$\arccos\left(-\frac{N \cdot E}{\| N \| \| E \|}\right) < \frac{\pi}{2} - \alpha, \tag{4}$$

then the area around the triangle Δ need not to be refined further, since after the refinement of Δ the corresponding triangles will also be on the backside of the object.

6.2 Improvement of the diffuse illumination near silhouettes

As shown in the introduction only small changes of the geometry in the area of silhouettes with respect to the light direction lead to broad artifacts of the resulting diffuse illumination of the object. Therefore, in areas of silhouettes we need to refine the triangulation until the projection of the triangles in screen space are small enough to be neglected. Assume again that the triangle Δ with normal N and normal deviation angle α and direction L to the light source is given. If

$$\frac{\pi}{2} - \alpha < \arccos(\frac{N \cdot L}{\| N \| \| L \|}) < \frac{\pi}{2} + \alpha, \tag{5}$$

then the area around the triangle need to be refined further, since in this case after further refinement of the geometry there might occur silhouettes in this area.

Let L be the normalized direction to the light source and E the normalized direction to the observer. The halfway-vector of the Blinn-Phong shading-model [1] used by OpenGL is defined by $H = \frac{L+E}{\|L+E\|}$. The luminance in this model is proportional to $(N \cdot H)^m$, where $m \geq 1$. We developed the following heuristic to detect a highlight: Assume that a triangle $\Delta = \Delta(v_1, v_2, v_3)$ with normal vector N, vertices v_1, v_2, v_2 and maximum normal deviation angle α is given.

(i) Ensure that neither the observer direction nor the light direction is on the backside of the triangle using the technique described in the section on back-face culling.

(ii) Calculate the halfway-vector. If a point-light source is used compute the halfway-vectors at the three corner vertices of the triangle H_1, H_2, H_3, $H = \frac{H_1+H_2+H+H_3}{\|H_1+H_2+H+H_3\|}$, and the maximum deviation angle β between H and $H_i,\ i = 1, 2, 3$.

(iii) If the angle between N and H

$$\angle(N, H) > \alpha + \beta + \gamma, \tag{6}$$

no highlight occurs in the area around the triangle and no further refinement is necessary. γ depends on the exponent m in the Blinn-Phong model and is normally an angle between 10° and 45°. It defines a range for the angle between surface normal and halfway-vector in which a highlight occurs.

6.4 Results

View-dependent triangulations can be extracted at interactive update-rates when specular illumination is not taken into consideration. For a realistic flight over the Grand Canyon area with about 700 km/h and a height between 700m and 5000m and an image resolution of 200×200 we achieved an average update rate of about 25 updates/sec on a on an Indigo with a 150MHz R4400 Processor. For details see [21].

Figure 14 shows results on illumination dependent extraction. We used the example of the car bonnet shown in the first section. The triangulation is only refined around the area of the highlight and along possible silhouettes. All artifacts described disappear. However, due to the refinement of the interesting areas the reduction rate decreases from about 95% in the triangulation shown in Figure 1 to about 60% in Figure 14. Unfortunately, the achievable update rates decrease even more than these numbers might suggest. The reason is,

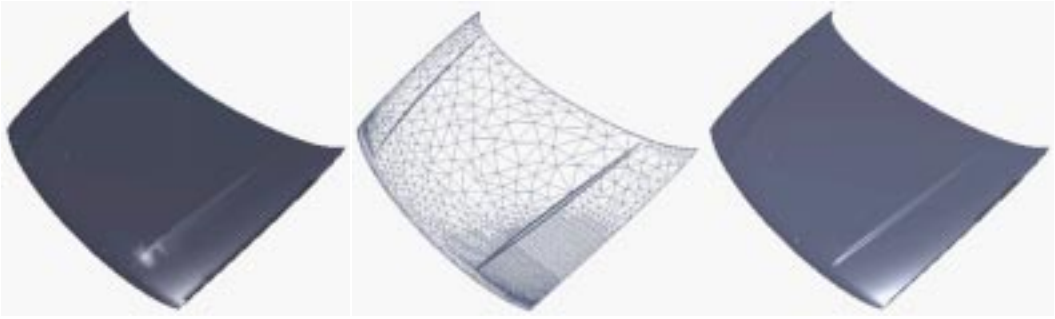

Fig. 14. The artifacts in the illumination of the car bonnet in the left picture are due to its coarse triangulation, see Figure 1. In the picture in the middle a triangulation with variable resolution is shown that takes into account the illumination of the bonnet. The resulting shaded bonnet is shown on the right. Note that not only the areas of the highlight are refined, but also the areas around silhouettes w.r.t. the light direction.

that we not only have to process more triangles per frame but that the number of triangles that have to be removed and inserted between two consecutive frames can be much higher. Highlights appear, disappear or move sometimes as a result of a slight movement of the observers position. The situation gets even worse, if we allow objects like the car bonnet to be rotated. Given our experience we doubt that in these situations other approaches described in the literature deliver better results.

6.5 Further applications

6.5.1 Interactive error tolerance editing

In many applications it is desirable to refine or to coarsen the mesh interactively in some areas, see for example Figure 5. In this example for the purpose of finite element computations the original mesh is refined around a single tooth where the maximum deformations during the computation are expected. To realize interactive editing the user can mark an area of triangles of the mesh through picking and assign smaller error tolerances to the triangles of the marked area. The algorithm then automatically refines the mesh until all triangles of the area fulfill the new error tolerances.

6.5.2 Progressive transmission

The proposed MRM-models built upon the mesh simplification algorithm also support progressive transmission. During the simplification process vertices are successively removed causing coarser and coarser meshes. The removal of vertices results in an increasing approximation error, although the removal

of a single vertex in some cases may produce even an intermediate result with smaller approximation error. For progressive transmission first of all the coarsest mesh, building the bottom of the MRM, is transmitted. Then, the vertices and in the case of the explicit model their accompanying fragments are transmitted in the reverse order of their removal. Once the bottom of the mesh is transmitted the receiving side can immediately start displaying it and then to refine the bottom mesh getting better and better approximations.

7 Conclusion and future work

The proposed multiresolution models both support several different levels of detail to co-exist across different regions of the object. The different levels of detail in different regions seamlessly merge with one another without introducing any cracks and discontinuities. The algorithms to refine and to coarsen a simplified mesh allow to exploit the frame to frame coherence between successive frames during the visualization process. Therefore, variable levels of detail can be extracted from the MRM in real-time, even for very complex meshes as long as specular illumination is not important. Nevertheless, taking into account the given illumination conditions high quality visualization is possible without refraining from mesh simplification. However, further work is required to achieve real-time update rates in these situations.

8 Acknowledgement

First of all I would like to thank A. Schilling for many fruitful discussions and hints and J. Krämer for all the programming efforts which were necessary to gain the described results. Thanks also to the anonymous reviewers for their thorough reviews and valuable hints.

References

[1] James F. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):192–198, July 1977.

[2] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. Technical Report CNUCE: C96-021, Istituto per l'Elaborazione dell'Informazione - Condsiglio Nazionale delle Richere, Pisa, ITALY, July 1996.

[3] P. Cignoni, C. Rochini, and R. Scopigno. Metro: measuring error on simplified surfaces. Technical Report B4-01-01-96, Istituto I.E.I. - C.N.R., Pisa, Italy, January 1996.

[4] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, October 1976.

[5] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Brooks, Jr., and William Wright. Simplification envelopes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[6] Leila de Floriani and Enrico Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4):363–411, October 1995.

[7] M. Eck, T. DeRose, T. D., H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 173–182. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[8] Carl Erikson. Polygonal simplification: An overview. Technical Report TR96-016, Department of Computer Science, University of North Carolina - Chapel Hill, February 16 1996. Wed, 26 Jun 1996 18:07:48 GMT.

[9] Tsung-Pao Fang and Les A. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics & Applications*, pages 36–47, may 1993.

[10] Tsung-Pao Fang and Les A. Piegl. Algorithm for Constrained Delaunay Triangulation. *The Visual Computer*, 10(5):255–265, 1994.

[11] L. De Floriani and E. Puppo. Constrained delaunay triangulation for multiresolution surface description. In *Ninth International Conference on Pattern Recognition (Rome, Italy, November 14–17, 1988)*, pages 566–569, Washington, DC, 1988. Computer Society Press.

[12] L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. In W. Straßer, R. Klein, and R. Rau, editors, *Theory and Practice of Geometric Modeling*. Springer-Verlag, 1996.

[13] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, August 1993.

[14] Charles Hansen and Paul Hinker. Isosurface extraction SIMD architectures. In *Visualization'92*, pages 1–21, oct 1992.

[15] P.S. Heckbert and M. Garland. Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1995.

[16] Ludger Hoffmann, editor. *OpenGL reference manual : the official reference document for OpenGL, release 1* . Addison-Wesley, Reading, Mass. u.a., 1992.

[17] H. Hoppe. Progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.

[18] H. Hoppe. View-dependent refinement of progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1997 (ACM SIGGRAPH '97 Proceedings)*, pages 189–198, 1997.

[19] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 19–26, August 1993.

[20] Alan D. Kalvin and Russel H. Taylor. Superfaces:polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Appl.*, 16(3), May 1996.

[21] R. Klein, D. Cohen-Or, and T. Hüttner. Incremental view-dependent multiresolution triangulation of terrain. In *Pacific Graphics '97 Proceedings*. IEEE Computer Society, IEEE Computer Society Press, 1997.

[22] R. Klein and S. Gumhold. Data compression of multi-resolution representations of surface meshes. submitted to the Hot Topics, Visualization 97, May 1997.

[23] R. Klein and T. Hüttner. Simple camera-dependent approximation of terrain surfaces for fast visualization and animation. In R. Yagel, editor, *Visualization 96*. ACM, November 1996.

[24] R. Klein and J. Krämer. Multiresolution representations for surface meshes. In W. Straßer, editor, *Proceedings of the Spring Conferece on Computer Graphics*, pages 57–66, June 1997. held in Budmerice, Slovakia, 05-08 June 1997.

[25] R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. In R. Yagel, editor, *Visualization 96*. ACM, November 1996.

[26] R. Klein and W. Straßer. Mesh generation from boundary models. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 431–440. ACM Press, May 1995.

[27] R. Klein and W. Straßer. Generation of multiresolution models from cad-data for real time rendering. In W. Straßer, R. Klein, and R. Rau, editors, *Theory and Practice of Geometric Modeling*. Springer-Verlag, 1996.

[28] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hughes, Nick Faust, and Gregory Turner. Real-Time, continuous level of detail rendering of height fields. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 109–118. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[29] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In R. D. Bergeron and A. E. Kaufman, editors, *Visualization '94 Proceedings*, pages 281–287. IEEE Computer Society, IEEE Computer Society Press, 1994.

[30] E. Puppo. Variable resolution of terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry*, August 1996.

[31] John Rohlf and James Helman. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[32] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):C67–C76, C462, September 1996.

[33] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scences. In B. Falcidieno and T. L. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465. Springer Verlag, 1993.

[34] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[35] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[36] J. C. Xia, J. El-Sana, and A. Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):171–183, 1987.

[37] J. C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In Holly Rushmeier, editor, *Visualization '96 Proceedings*, volume 30(4), August 1996.